Phylogenetic Tree Parsing with Stack-based Data Structure for IBD Detection, and Algorithm's Upgrade

Shuo Yang

Pe'er lab, Columbia University

New York, USA

May.23 2014

Outline

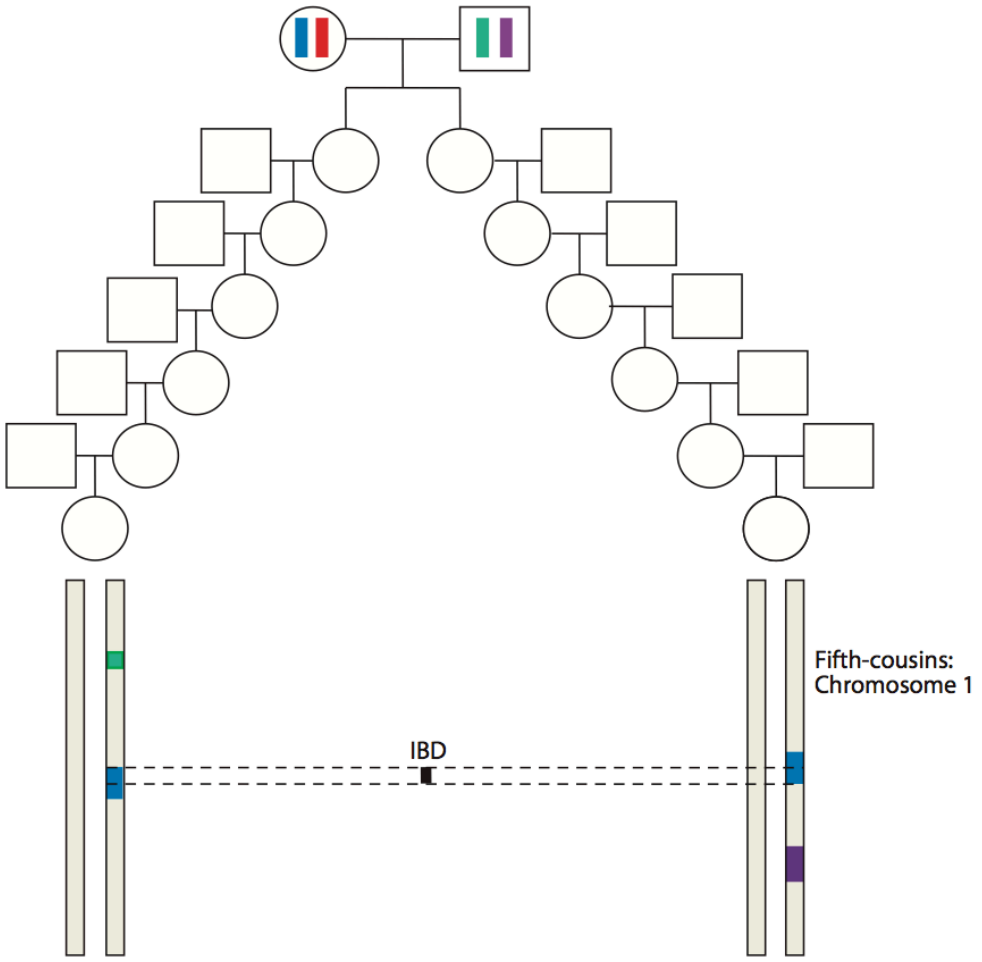
- Problem Description
- Naïve Method Implementation
- Algorithm Upgrade and Implementations Section 1 (direct tMRCA report)
- Algorithm Upgrade and Implementations Section 2 (candidate based; undergoing)
- Summary

Outline

- Problem Description
- Naïve Method Implementation
- Algorithm Upgrade and Implementations Section 1 (direct tMRCA report)
- Algorithm Upgrade and Implementations Section 2 (candidate based; undergoing)
- Summary

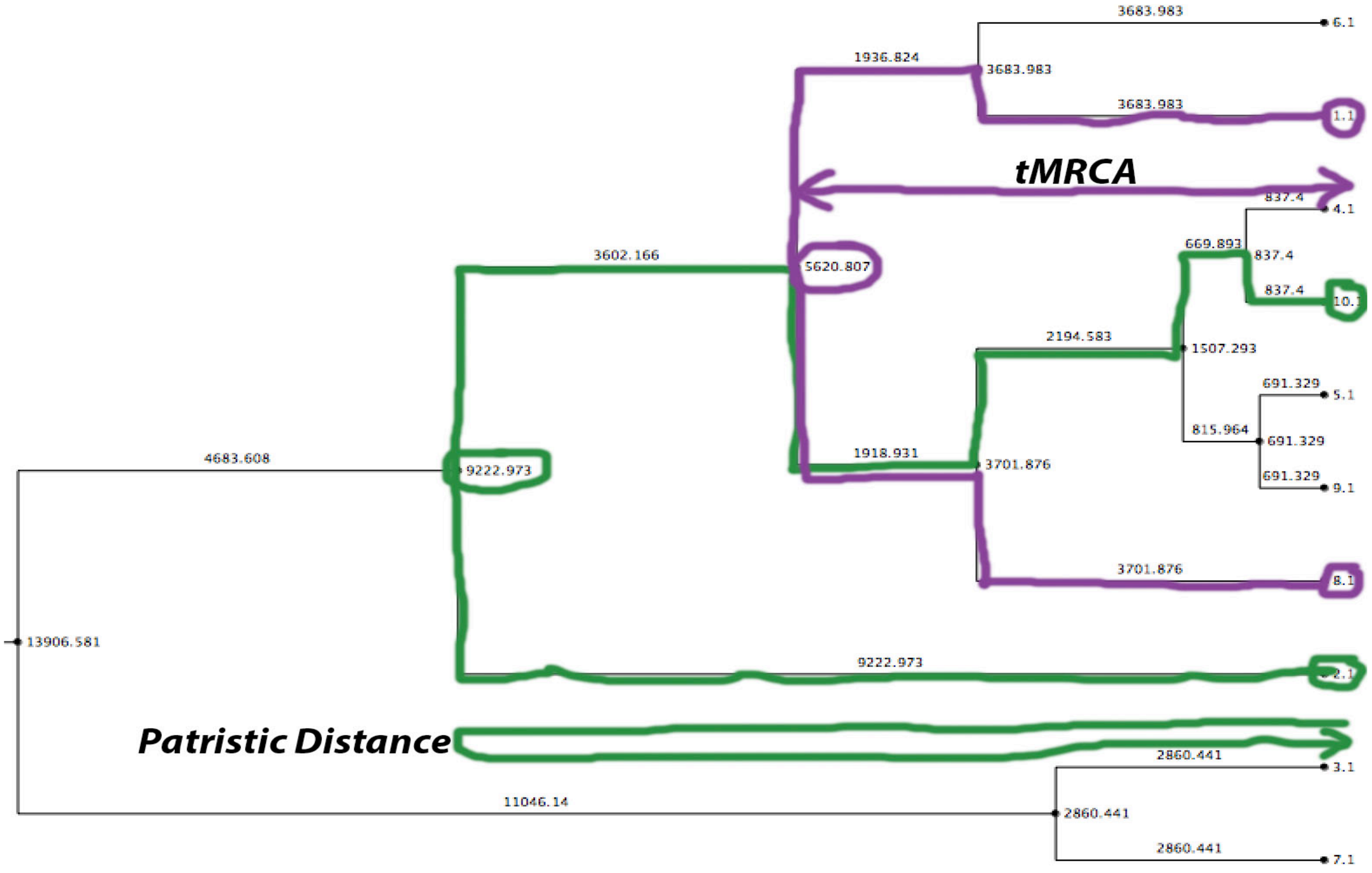
Identity by Descent (IBD)

- **Identity by descent (IBD):** two alleles or haplotypes that are identical and are inherited from a shared ancestor
- **IBD segment:** a continuous segment over which two haplotypes are identical by descent

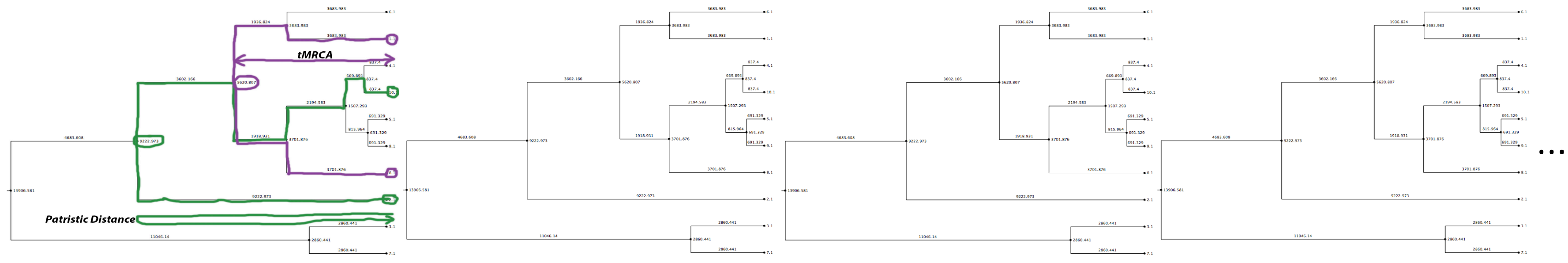


Sharon R. Browning and Brian L. Browning. 2012. *Identity by Descent Between Distant Relatives: Detection and Applications*. Annu. Rev. Genet. 46:617–33

IBD Extraction from Phylogenetic Trees



IBD Extraction from Phylogenetic Trees



tMRCA / range / pair	1	540	1120	2010	2630 ...
1-2:	125	✗	125	231	245 ...
1-3:	112	132	✗ 132	154	...
2-5:	121	✓ 121	121	245	...
...	...	...	...	...	...

Comparison-based Algorithm Time Complexity

$$\text{tree\#} \approx 2NL \log n$$

$$running\ time: O\big((tree\#)n^2\big) = O(NL \underbrace{n^2}_{\substack{potential\\ improvement}} \log n)$$

$$N = population\ size$$

$$L = the\ length\ of\ the\ chromosome$$

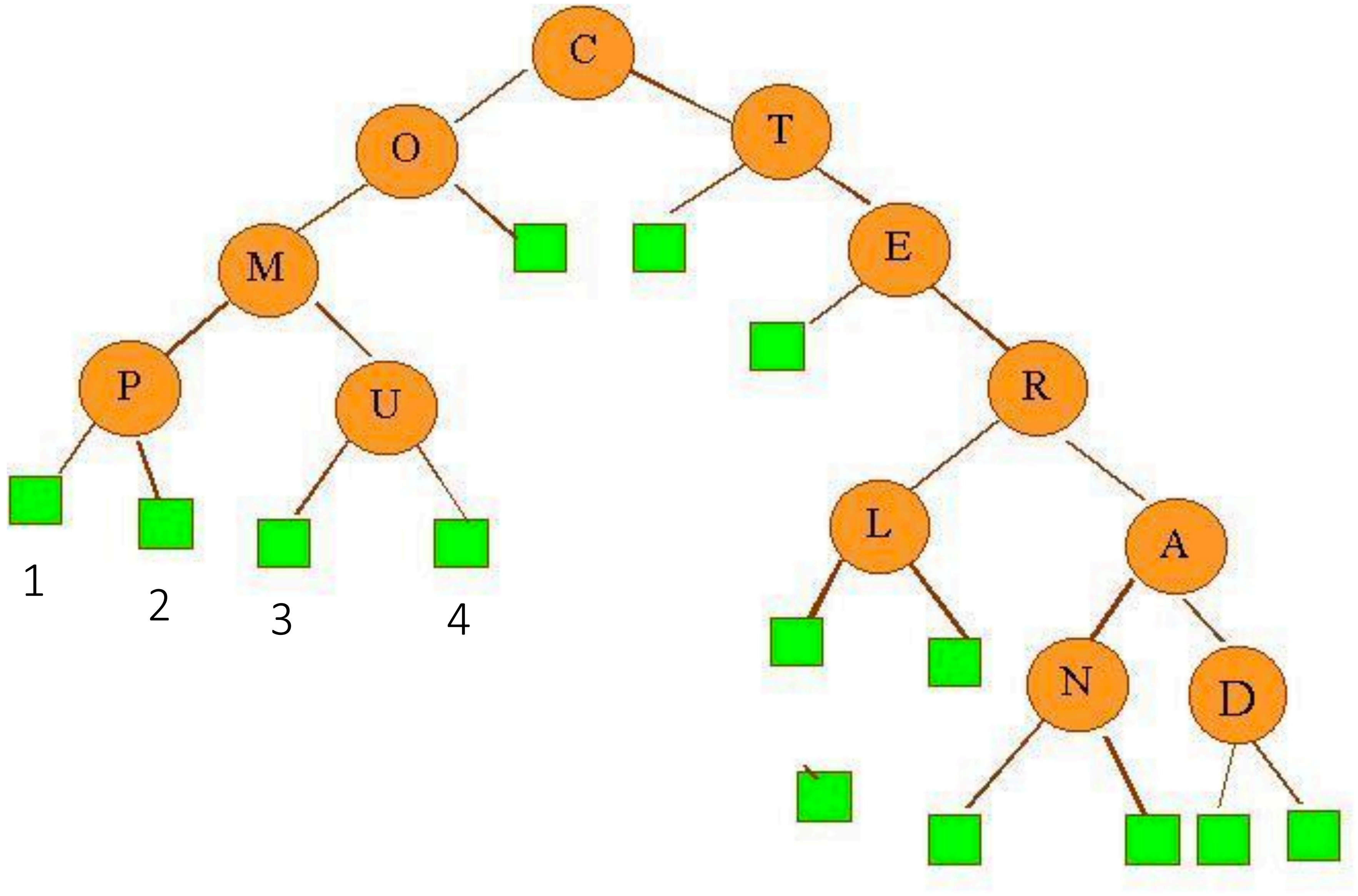
$$n = number\ of\ simulated\ chromosomes$$

* from Shai

Outline

- Problem Description
- Naïve Method Implementation
- Algorithm Upgrade and Implementations Section 1 (direct tMRCA report)
- Algorithm Upgrade and Implementations Section 2 (candidate based; undergoing)
- Summary

Report tMRCA while Tree Traversal



Implementation#1-1: Objected-oriented model

- Design a class for the Node object
- Parse the tree into indexed Node object
- Perform the tree traversal (in-order depth-first search); report tMRCA for each pair ever seen
- Developed in Python

Finishing time: Feb.20

Implementation#1-2: Using open source package (DendroPy)

- Back to the "first step"
- Calculate the patristic distance of each pair
- A very simple implementation, like a "code version" of the algorithm
- Nearly the same speed as before, demonstrating the drawback of objective-oriented method in modeling this problem

Finishing time: Mar.2

Outline

- Problem Description
- Naïve Method Implementation
- Algorithm Upgrade and Implementations Section 1 (direct tMRCA report)
- Algorithm Upgrade and Implementations Section 2 (candidate segment based; undergoing)
- Summary

Why we are bothering to build the tree model first?

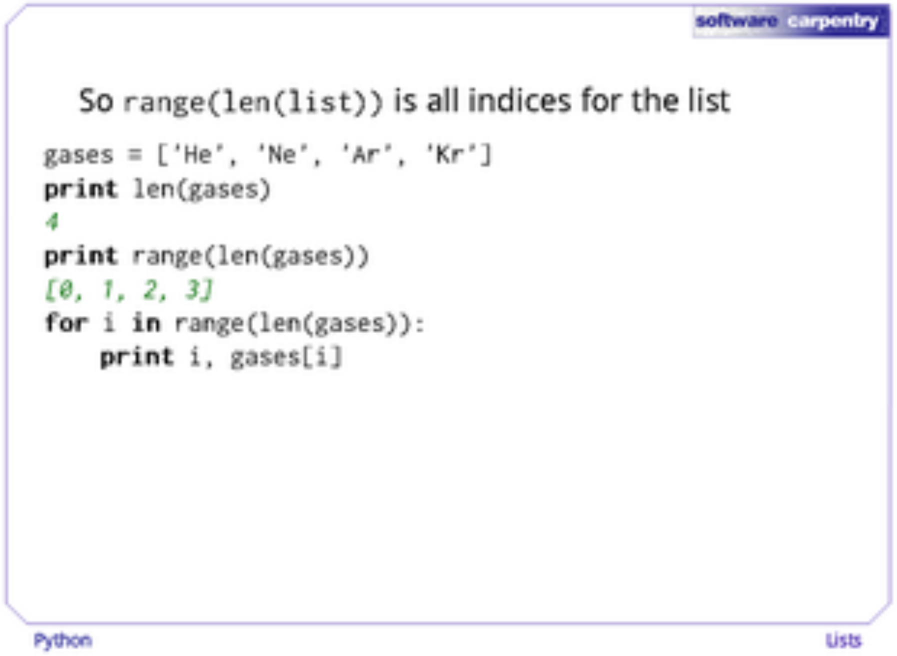
Observation:

1. We can get all-pairs tMRCA in one tree if and only if we traverse the tree;
2. The Nexus format of a tree perfectly describes the whole process of in-order depth-first traversal of a tree.

We detect/report tMRCA while parsing the tree, other than model the tree first

Implementation#2-1: Using Python to try this idea

- Stack-based data structure, directly report tMRCA when they are ready, no extra time needed, pure n^2 time spent
- Use *List* to simulate the behavior of a stack
- 10 times gain by now, relative to the previous implementations and the Matlab's implementation
- Finally even becomes the soul of all implementations



So range(len(list)) is all indices for the list

```
gases = ['He', 'Ne', 'Ar', 'Kr']
print len(gases)
4
print range(len(gases))
[0, 1, 2, 3]
for i in range(len(gases)):
    print i, gases[i]
```

Finishing time: Mar.6

So, what's next to become faster?

- Maybe a more subtle algorithm need some time to appear; but at least we have not yet tried all the programming language weapons
- Let's try it with C!

List in Python:
The good thing is sometimes the bad; you can serve all, but you are not specific enough for me; but I understand, because on one is perfect

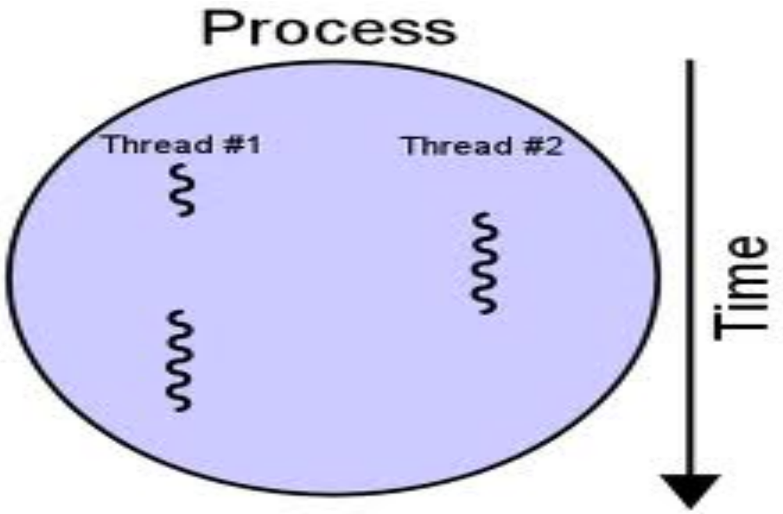
Thank Dennis, we can define ***structure*** in C, and we have the sharp ***pointers***, and we are the managers of ***memory***

Implementation#2-2: Using C with multi-threads technique

- Build a Python similar “list” in C to model the stack, but only code the necessary things for our task
- Carefully manage the memory (allocation and access – RAM consumption and running speed)
- Carefully design the parallel program; divide all the trees into several chunks, and merge the boundaries finally
- Another 10 times gain by now (without multi-threads)

```
#include <stdio.h>

int main()
{
    printf("Hello World");
    return 42;
}
```



Finishing time: Apr.11

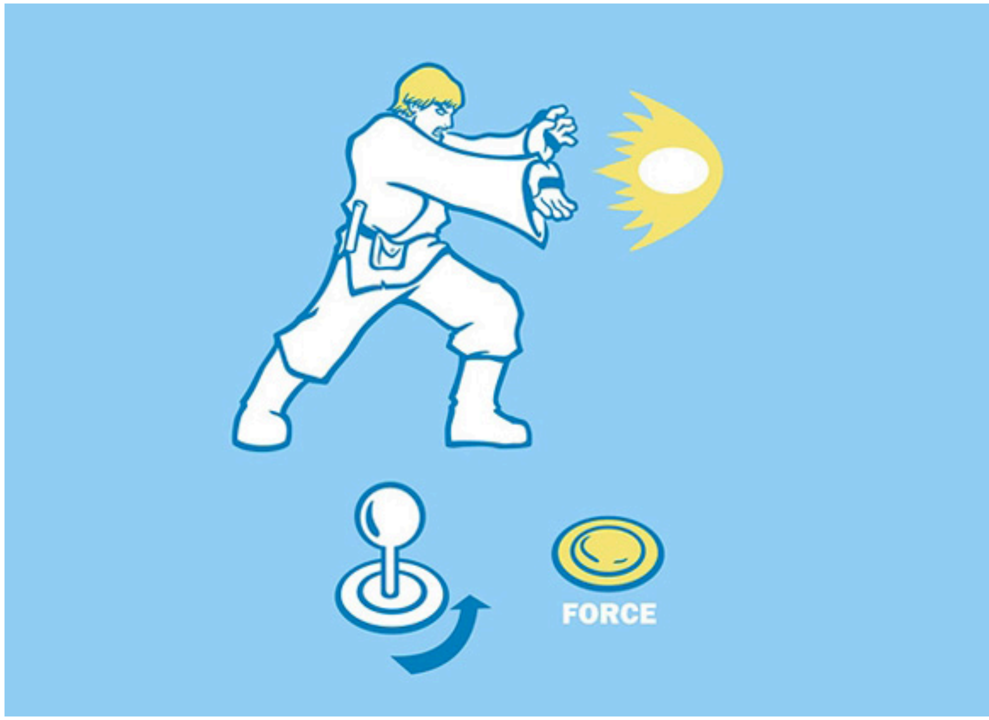
Implementation#2-3: Using C with multi-processes technique (OpenMPI)

- Fully use the recourse of our computation facilities – C2B2 clusters
- Deploy all the sub-jobs (chunks) to different processes in different CPU nodes, other than different threads in one process
- Save the boundaries of all tree chunks in temporary files in disk
- Overall working time depends on the size of one task, because of the extra consumption of dividing and merging

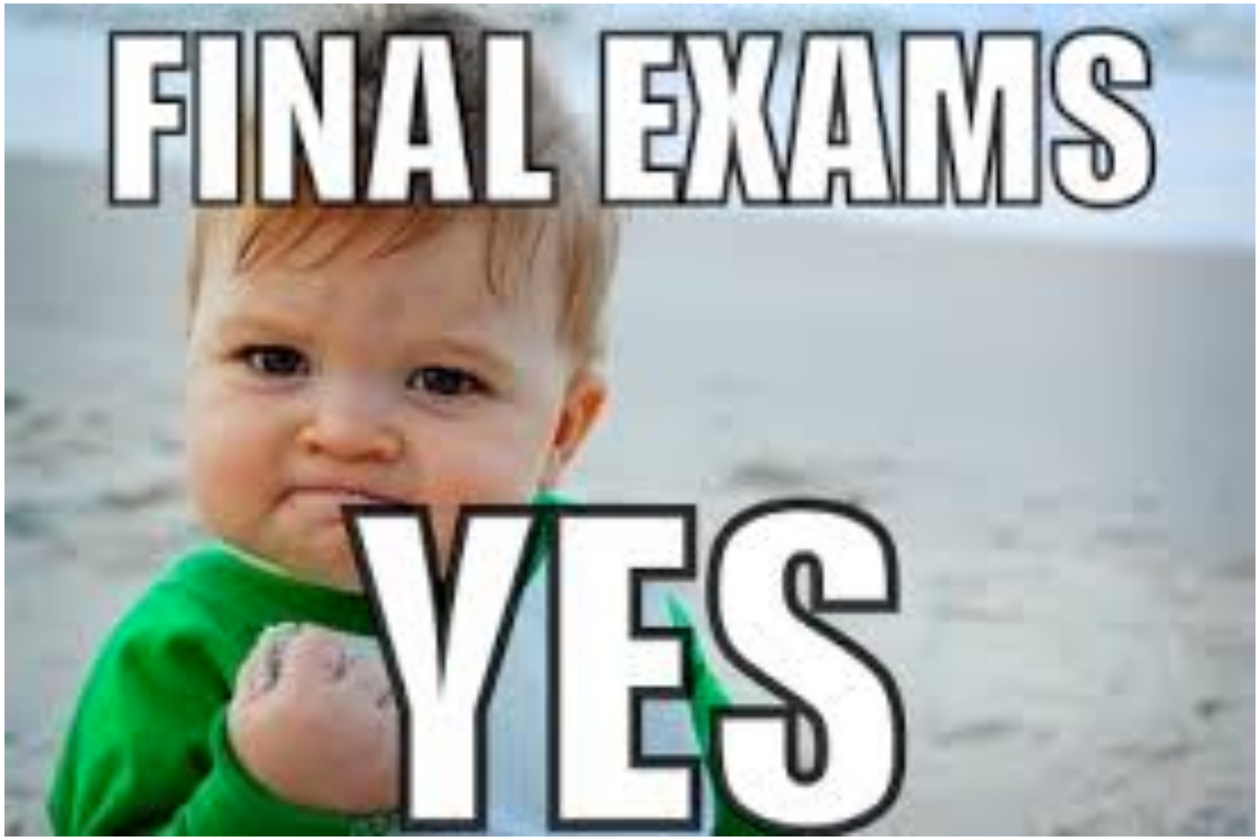
Finishing time: Apr.18

Wrapping up

- Make some parameters changeable by users when invoking the program – tree file name; format of tree; input type (file or stdin); cutoff value; discretization value; length of chromosome; number of working threads (processes); epsilon value (the tolerance for the tMRCA changes)
- Make the source code public



Finishing time: Apr.30



May.1 – May.15

Outline

- Problem Description
- Naïve Method Implementation
- Algorithm Upgrade and Implementations Section 1 (direct tMRCA report)
- Algorithm Upgrade and Implementations Section 2 (candidate based; undergoing)
- Summary

Want to be more subtle in Algorithm?

Recall the time complexity:

$$\text{tree\#} \approx 2NL\log n$$

$$running\ time: O\big((tree\#)n^2\big) = O(NL \underbrace{n^2}_{\substack{potential\\ improvement}} \log n)$$

$$N = population\ size$$

$$L = the\ length\ of\ the\ chromosome$$

$$n = number\ of\ simulated\ chromosomes$$

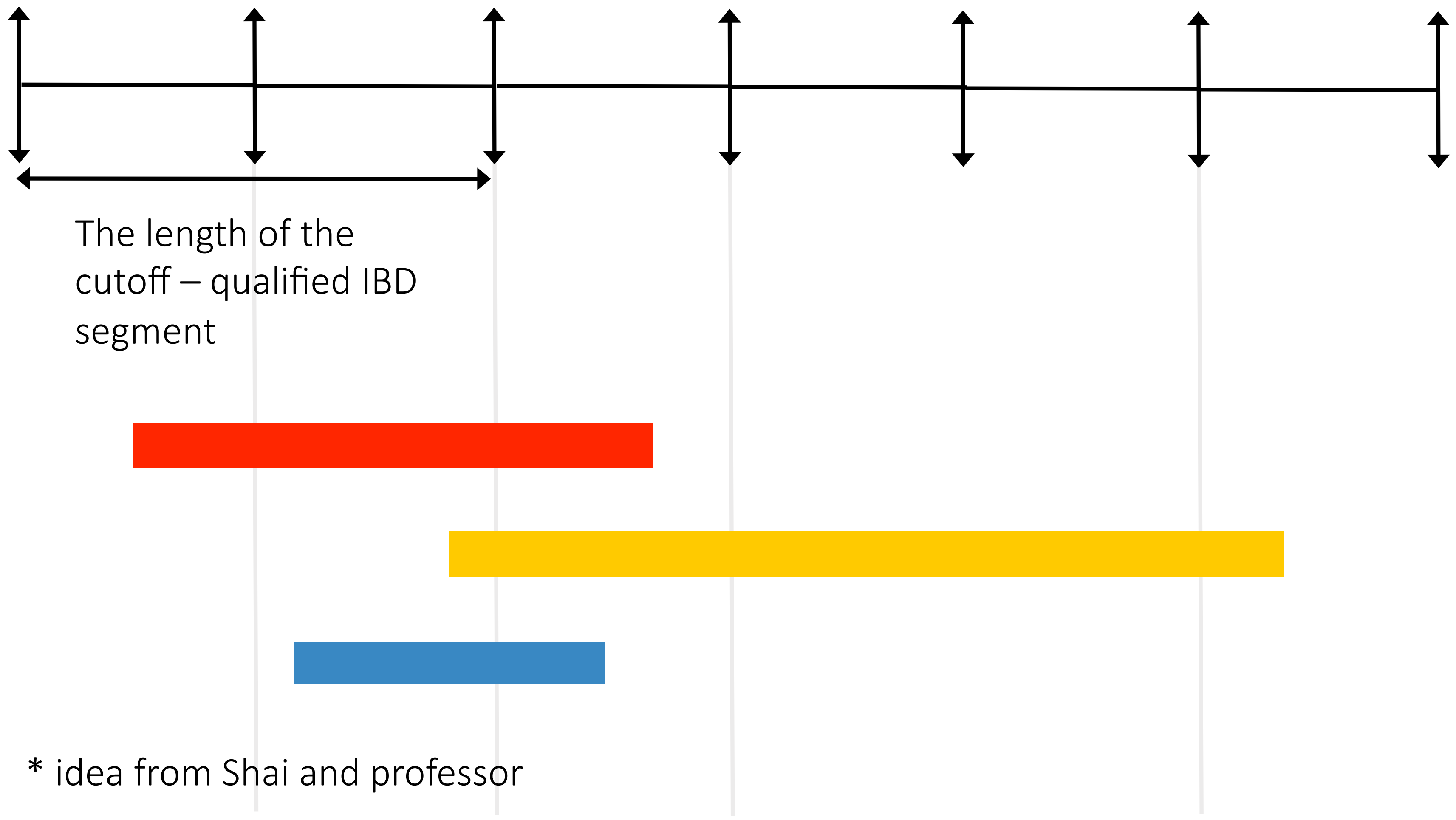
n^2 is the breakthrough point

two possible directions:

1. detect the tiny changes between two trees
2. get some IBD segment candidate first

A Subtle Candidate-based Algorithm

Observation:



* idea from Shai and professor

A Subtle Candidate-based Algorithm

- Use half-cutoff discretization to extract all-pairs IBD (actually making the total length of the chromosome several chunks)
- For each chunk, detect which pair covers this range (no tMRCA changes between the beginning of the chunk and the ending of the chunk), and record them as candidates
- For each chunk, verify all the pairs who are candidates of previous chunk/present chunk/next chunk; as there are less pairs than before, we expect doing this in constant time O(n) for one tree to extract their tMRCA
- Should be very fast, but more appropriate for large cutoff value
- Have finished 440 lines C, maybe 1/3 of the whole project; expect to finish in one week
- Eager to see the practical gain in running speed

* idea from Shai and professor

Outline

- Problem Description
- Naïve Method Implementation
- Algorithm Upgrade and Implementations Section 1 (direct tMRCA report)
- Algorithm Upgrade and Implementations Section 2 (candidate based; undergoing)
- Summary

Summary

- Try naive method and new algorithm and different types of programming tools for the task
- Design a sharp tool for other projects
- Get a sense of what I will do during the summer
- The most important thing for me: get enjoyments during the whole process, although there are hard times

Special thanks

- Thank Shai for the guidance; sometimes directions are as important as what you are eager to do
- Thank professor for the opportunity; sometimes opportunity is as important as what you can do
- Thank C2B2 staff for patient answering for some low-level questions
- Let's see what will happen in the near future!